Pallas: HPC Trace analysis at scale

Catherine Guelque[•], Philippe Swartvagher[▲], Valentin Honoré[•■], François Trahay[•]
• Télécom SudParis, Institut Polytechnique de Paris, Inria Benagil
▲ Bordeaux INP, Inria Topal
■ ENSIIE
Presented at IPDPS25 (hal-04970114)

gitlab.inria.fr/pallas/pallas

Context: performance analysis at exascale

Investigating performance problems

- Many sources of performance problems at scale
  - Collective communication, load imbalance, network contention, NUMA effects, ...
- Finding a bottleneck requires investigation
  - Run the application once, analyze its execution trace
- Applications mix programming models (MPI+OpenMP, MPI+CUDA), and libraries (StarPU, netcdf, Pytorch, ...)
- Need for **generic, modulable tracing tools**

Analyzing huge performance data

Finding the needle in the haystack

- Traces are **records of execution** made for **post-mortem analysis**
  - Store vast amount of detailed information
  - Require a lot of processing before being useful
- Trace analysis ressources scale with trace size
  - Gets worse with execution time and # of threads
- HPC applications have **recurring patterns**
  - If leveraged, would yield **fast, scalable analysis**

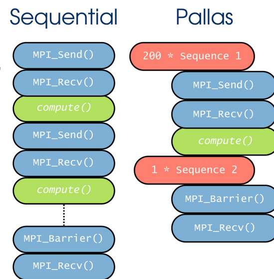
Proposal: Pallas trace format

Detecting repetitions at runtime

- Functions are **grouped by call signature** as Tokens.
- **Repetition of Tokens** are **stored as loops**
- **Durations** and **timestamps** are stored in buckets with **statistics** for **fast data retrieval**

A new trace format ?

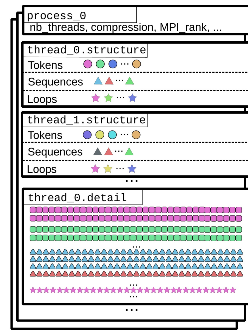
- Fully **generic** trace format
- **OTF2** and **Python** API for compatibility
- Set of basic analyses provided
  - Communication matrix
  - Contention[1] detection
  - Histogram plotting
  - Visualisation



Analysis-optimized trace storage

Separating data from metadata

- One folder per thread
- Header file with general information
  - Grammar / Function names
  - Runtime statistics
- Specialized files contain analysis data
  - Easily removable
  - On-demand loading



Evaluations

Comparison between:

- EZTrace / OTF2 (no pattern detection)
- EZTrace / Pallas (with our OTF2 API)
- Pilgrim[2] (perfect pattern detection)

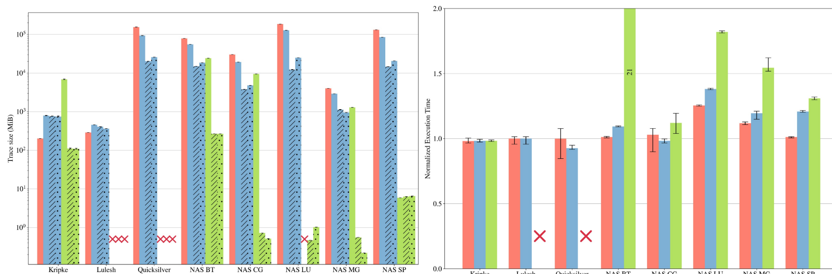
Ran 5x, **4096 MPI nodes** (Jean-Zay, Idris)

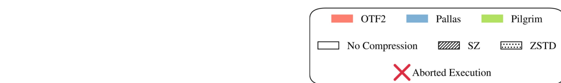
Trace size:

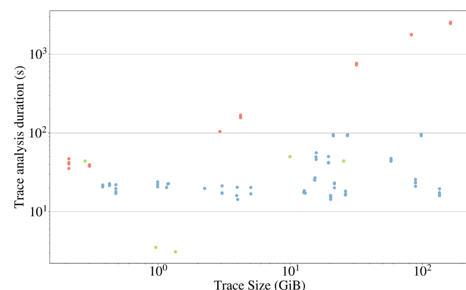
- OTF2 generates the biggest traces
- Pallas stores more information than Pilgrim
- Pilgrim has great compression
  - **Tends to crash due to OOM**
  - Needs to decompress everything when reading

Runtime overhead:

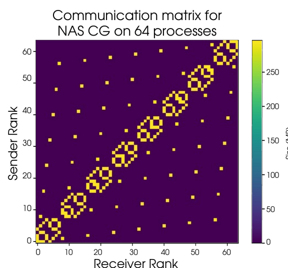
- OTF2 and Pallas both have **low overhead**
- Pilgrim adds **significant** overhead







Communication matrix for NAS CG on 64 processes

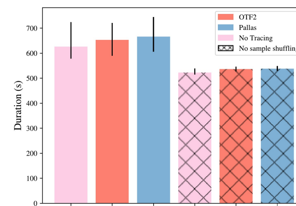


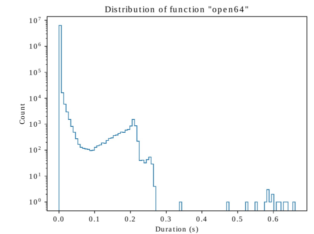
Communication matrix plotting:

- OTF2 scales linearly with trace size
  - The whole trace needs to be parsed
- Pallas and Pilgrim have **near-constant analysis time**
  - Only the grammar is read

Case-study: ResNet50 / ImageNet

- Deep-learning application with known[3] **I/O contention issue**
- **128 MPI processes** on the GPU partition of Jean-Zay
- Traced **MPI, I/O and Python** functions using EZTrace
  - Compared OTF2 and Pallas
  - **Pilgrim does not provide a proper reading API**
- Analysis process in 3 steps:
  - Identify the most time-consuming functions
  - Histogram these functions to identify issues
  - Compute a contention score





Metric	OTF2	Pallas
Performance overhead	4.2 %	6.4 %
Number of collected events	428,415,180	429,008,944
Number of files	21,003	42,255
Total trace size	6.1 GiB	1.4 GiB
Time to profile	40.46 s	8.37 s
Time to compute histograms	1285.34 s	144.67 s
Time to estimate contention	40.46 s	7.93 s

Ongoing work

- Improve analysis API (Python, R)
- Improve tracing performance, trace visualization
- Scaling :
  - *Horizontally:* Longer execution times
  - *Vertically:* More threads / processes
  - *Analysis:* TBs of data in a few seconds

1: M. S. M. Bouksiaa, F. Trahay, A. Lescouet, G. Voron, R. Dulong, A. Guermouche, E. Brunet, and G. Thomas, "**Using differential execution analysis to identify thread interference**"
2: C. Wang, P. Balaji, and M. Snir, "**Pilgrim: Scalable and (near) Lossless MPI Tracing**"
3: T. T. Nguyen, F. Trahay, J. Domke, A. Drozd, E. Vatai, J. Liao, M. Wahib, and B. Gerofi, "**Why globally re-shuffle? Revisiting data shuffling in large scale deep learning**"

Contact: catherine.guelque@telecom-sudparis.eu